Rapport Reconnaissance faciale

CASSISI Ryan
IA et Deep Learning
ESIEE - 2023
France
ryan.cassisi@edu.esiee.fr

BOUGHEMZA Hafsa
IA et Deep Learning
ESIEE - 2023
France
hafsa.boughemza@edu.esiee.fr

Abstract

Le deep learning est l'une des méthodes les plus performantes pour la reconnaissance faciale. En effet, des réseaux de neurones profonds permettent d'extraire des caractéristiques de plus en plus abstraites des images de visages.
Cependant, l'un des problèmes majeurs et les plus récurrents du deep learning peut venir entraver les performances de notre modèle : l'Overfitting.
Pour y faire face, un pré-traitement des données est primordiaI. Il se décompose en 3 étapes : La détection de visage, le recadrage des visages, et l'encodage des visages.
Ce prétraitement est essentiel pour améliorer la précision du modèle.
Seulement après avoir réalisé ces étapes, nous pouvons entraîner notre réseau de neurones afin d'associer un nom à un nouveau visage.

Index

Deep learning - Apprentissage supervisé - Classification - Reconnaissance faciale - Convolution layer

Introduction

Le deep learning est l'une des méthodes les plus performantes pour la reconnaissance faciale. En effet, des réseaux de neurones profonds permettent d'extraire des caractéristiques de plus en plus abstraites des images de visages.
Cependant, l'un des problèmes majeurs et les plus récurrents du deep learning peut venir entraver les performances de notre modèle : l'Overfitting.
Pour y faire face, un pré-traitement des données est primordiaI. Il se décompose en 3 étapes : La détection de visage, le recadrage des visages, et l'encodage des visages.
Ce prétraitement est essentiel pour améliorer la précision du modèle.
Seulement après avoir réalisé ces étapes, nous pouvons entraîner notre réseau de neurones afin d'associer un nom à un nouveau visage.

Related Work

Le premier travail est celui de Matthew Turk et Alex Pentland , intitulé "Eigenfaces for Recognition" [1]. Ce travail utilise une méthode traditionnelle de reconnaissance des visages appelée Eigenfaces. Ils ont utilisé une base de données qui contient 400 images de 40 personnes différentes. Les images de visages collectées ont un large éventail de conditions d'imagerie. La méthode Eigenfaces utilise l'analyse en composantes principales (PCA) pour extraire les caractéristiques des visages et créer une représentation en 2D de chaque visage. Les résultats étaient bons, avec une précision de reconnaissance de plus de 95 %. Cependant, la méthode Eigenfaces n'est pas très robuste aux variations d'illumination et de pose.

Le dernier travail est intitulé "DeepFace: Closing the Gap to Human-Level Performance in Face Verification" de Yaniv Taigman, Marc'Aurelio Ranzato, Lior Wolf et Ming Yang [3]. Ce travail utilise une approche d'apprentissage profond (deep learning) pour la reconnaissance des visages. Ils ont utilisé une base de données interne de Facebook, qui contient plus de 4 millions d'images de plus de 4 000 personnes différentes. Leur modèle utilise un réseau de neurones convolutionnels pour extraire les caractéristiques des visages, suivi d'un réseau de neurones entièrement connecté pour la classification. Les résultats étaient exceptionnels, avec une précision de reconnaissance de plus de 97 % sur la base de données LFW (Labeled Faces in the Wild). Cependant, ce travail a été critiqué pour son manque de transparence et de données de test publiques. De plus, des données de test publiques pourraient être fournies pour permettre une comparaison plus équitable avec d'autres travaux.

Proposition

A - Face Detection

La détection faciale consiste, à partir d'une image, à localiser les visages en faisant appel à des algorithmes de détection de visage. C'est la première étape dans une reconnaissance faciale. [4]

Les données que nous avons reçues pour la détection de visage sont des images de 6 personnages de Jurassic Park : Alan Grant, Claire Dearing, Ellie Sattler, Ian Malcolm, John Hammond et Owen Grady.
Chaque personnage ayant 35 images différentes.
Nous devons d'abord détecter les visages sur ces données car nous analysons seulement les visages, et non l'image entière avec son arrière-plan, pour les catégoriser. La première étape est donc de localiser les visages sur une photo.

Nous avons pré-traité nos données en commençant par la localisation des visages sur chacune des images, puis l'extraction de ceux-ci en tant que nouvelle image. Après ce prétraitement, toutes les images ont la même dimension (128x128) et contiennent uniquement un visage.
Les labels à associer à chaque visage ont également été extraits.
Pour réaliser ce prétraitement, nous avons créé deux listes : faces et labels
Celles-ci ont été remplies en utilisant la fonction extract_faces et en conservant le nom du personnage correspondant présent dans le chemin d'accès de l'image.
Les visages et labels ont ensuite été stockés dans un fichier grâce à la fonction dump du package pickle.
Afin que nos données soient plus efficacement traitées par nos modèles de deep learning, nous les avons normalisées en divisant chaque pixels par 255.

Image avant prétraitement

Image après prétraitement

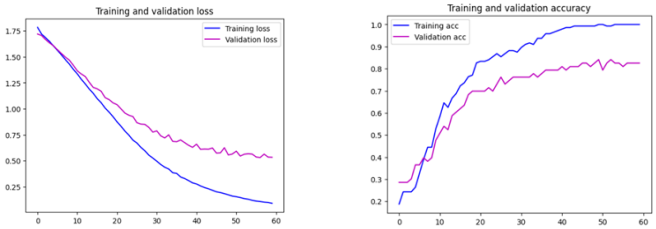
Pour entraîner et évaluer notre modèle de prédiction, nous avons divisé nos données en deux parties :

- Training set (70%) : les données destinées à l'apprentissage
- Test set (30%) : les données destinées à l'évaluation

Pour cela nous avons utilisé la fonction train_test_split de sklearn. Les données ont été mélangées avant cette séparation afin de s'assurer que tous les personnages soit présent dans les deux ensembles. (paramètre shuffle=True)

Lorsque les entrées sont des images, on utilise un ConvNet, qui est une forme spéciale du réseau neuronal artificiel et qui est composé de plusieurs couches de convolution. Le ConvNet permet alors d'encoder certaines propriétés dans l'architecture, rendant la propagation vers l'avant plus efficace et reduisant le nombre de paramètres.
Notre ConvNet est composé de 2 layers SeparableConv2D, 2 layers de MaxPooling2D, 1 layer Flatten et 2 layers Dense.
Le SeparableConv2D réduit le nombre de paramètres de poids entraînables et d'opérations en virgule flottante, ce qui se traduit par des modèles plus légers et plus rapides. Il sépare l'apprentissage des caractéristiques spatiales et des caractéristiques par canal, ce qui le rend plus efficace et nécessite moins de données pour obtenir de meilleures performances. Il est particulièrement utile pour les petits modèles formés sur des données limitées comme le notre.
Input_shape est la taille des images en entrée, soit (128,128,3).
Les piles de couches "denses" avec des activations "relu" résoudront un large éventail de problèmes, c'est pour cela qu'on les utilisent fréquemment.
Comme nous avons ici un problème de classification multiclasse (6 classes en sortie), notre réseau se termine par une couche 'Dense' avec 6 unités et une activation 'softmax', c'est-à-dire que la sortie de notre réseau devrait être une distribution de probabilité. Le fonction de perte que nous avons utilisé est la 'categorical_crossentropy' car notre modèle à 6 résultats possible.
Nous avons mis 60 epochs, afin que le modèle puisse converger sans overfit.
Le steps_per_epoch est déterminé automatiquement.

Nous obtenons ces graphiques :



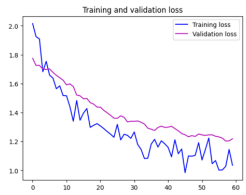

```
Epoch 60/60
4/4 [==============================] - 0s 37ms/step - loss: 0.0910 - accuracy: 1.0000 - val_loss: 0.5338 - val_accuracy: 0.8254
```

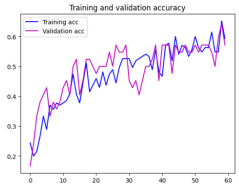
A partir du graphique, on observe de l'overfitting : la courbe de la perte d'entraînement continue de diminuer lorsque que la courbe de la perte de validation commence à stagner.

Afin de pallier ce problème d'overfitting et améliorer les performances du ConvNet, nous utilisons la technique de data augmentation. Pour ce faire, un générateur d'images déformant les images doit être mis en place pour augmenter les données.
Afin d'améliorer les performances du convnet de base, nous ajoutons un generateur d'image afin d'augmenter les données.
Nous avons utilisé ce qui a été vu au TP5 : using convNet with small data

De plus, nous avons ajouté des dropouts layers ainsi que la L2-norm penalization pour réduire davantage l'overfitting.

Nous obtenons ces graphiques :






```
Epoch 60/60
7/7 [==============================] - 1s 150ms/step - loss: 1.0984 - accuracy: 0.5926 - val_loss: 1.2554 - val_accuracy: 0.5714
```

Nous voyons que la méthode de data augmentation et l'ajout de nos dropout layers réduit le problème d'overfitting. Cependant, la précision du modèle n'est pas pour autant meilleure.
L'ajout de layer et la modification de leur nombre de filtre n'améliore pas le modèle.
Nous ne comprenons pas non plus ce qui est à l'origine de l'oscillation des courbes dans les graphiques.

B - Face Detection

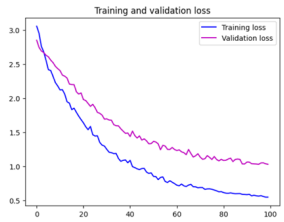
La pose estimation est essentielle afin d'avoir à un algorithme précis. Celle-ci consiste à aligner la représentation des visages en détectant et suivant la position et l'orientation de parties du corps humain dans des images, selon un protocole spécifique afin de faciliter la reconnaissance[5].

Dans la première partie nous avons extrait les visages mais nous faisons maintenant face au problème suivant : les visages tournés dans des directions différentes ont un aspect totalement différent pour un ordinateur.
Dans cette partie, nous avons prétraité davantage les images de visages en corrigeant leur pose afin d'obtenir des visages recadrés, alignés et normalisés.

Nous devons alors déformer chaque image de manière à ce que les yeux et les lèvres se trouvent toujours au même endroit dans l'image.
Nous avons utilisé l'algorithme landmark estimation.
Nous voulons localiser 68 points spécifiques (landermarks) qui existent sur chaque visage (le haut du menton, le bord extérieur de chaque œil, le bord intérieur de chaque sourcil…).
Par la suite, nous mettons à l'échelle et nous cisaillons l'image afin que les yeux et la bouche soient centrés le mieux possible.
La reconnaissance des visages sera ainsi plus précise.

Cet alignement a été géré par les fonctions fournies. Les données ont d'abord été récupérées via le package pickle puis normalisées et alignées avec les fonctions fournies pour ensuite être utilisées dans le modèle.

Nous avons divisé ces nouvelles données en deux ensembles comme précédemment (Training set et Test set) afin d'entraîner et évaluer notre ConvNet.




```
100/100
==============================] - 0s 42ms/step - loss: 0.6334 - accuracy: 0.9931 - val_loss: 1.5833 - val_accuracy: 0
```

Nous observons toujours de l'overfitting, cependant les valeurs de précisions sont meilleures. Cela s'explique par le fait que comme les visages ont été recadrés, ils sont rendus plus exploitables

C - Face Encoding

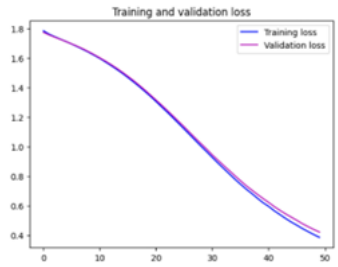
Classer directement un visage inconnu à l'aide d'un convnet formé sur notre base de données de personnes étiquetées, semble être la manière la plus simple. Cependant, entraîner un tel convnet sur une importante base de données prendrait beaucoup trop de temps.
Il faut donc trouver un moyen d'extraire quelques mesures de base de chaque visage, que l'on pourra ensuite utiliser pour comparer rapidement le visage inconnu à notre base de données.
Les chercheurs ont découvert que l'approche la plus précise consiste à laisser l'ordinateur déterminer lui-même les mesures à collecter. L'apprentissage profond est plus efficace que les humains pour déterminer les parties d'un visage qu'il est important de mesurer.
L'algorithme note donc certaines mesures importantes sur le visage, comme la couleur, la taille et l'inclinaison des yeux, l'écart entre les sourcils, etc. Tous ces éléments réunis définissent l'encodage du visage: l'information obtenue à partir de l'image qui est utilisée pour identifier le visage en question [6].

Le but de cette partie est donc de former un convnet et de l'entraîner à générer 128 mesures pour chaque visage.

Nous avons tout d'abord prétraiter les visages recadrés en les encodant avec la fonction face_encoder fournie dans le TP.
Nous entraînons ensuite notre modèle sur ces nouvelles données.
Comme les visages encodés ne sont que des vecteurs de longueurs 128, nous n'avons pas besoin d'un convnet, mais d'un réseau ordinaire avec une série de couches entièrement connectées.
Le modèle construit se constitue alors seulement de layers Dense.
Le modèle est composé de :

```
model_encoded = models.Sequential()
model_encoded.add(layers.Dense(512, activation='relu', input_shape=(128,)))
model_encoded.add(layers.Dense(128, activation='relu'))
model_encoded.add(layers.Dense(64, activation='relu'))
model_encoded.add(layers.Dense(6, activation='softmax'))
model_encoded.compile(loss='categorical_crossentropy', optimizer=optimizers.Adam(l
```

Nous avons ainsi évaluer les performances sur l'ensemble de test:



Le modèle n'étant composé que de layer dense, il converge beaucoup plus vite et est beaucoup plus performant. Le modèle atteint une précision entre 96% et 99%.
Ses performances sont meilleures que les modèles créés auparavant.

D - Face recognition

Cette partie consiste à trouver la personne dans notre base de données de personnes connues qui a les mesures les plus proches d'une image test.
Pour ce faire, nous avons utilisé plusieurs algorithmes de classification par apprentissage automatique : svm, knn et linear model.

L'algorithme des k plus proches voisins (KNN) est un discriminant d'apprentissage supervisé non paramétrique, qui utilise la proximité pour effectuer des classifications ou des prédictions sur le regroupement d'un point de données individuel. [7]

L'algorithme SVM (Support Vector Machine) est une technique d'apprentissage supervisé utilisée pour la classification et la régression. L'objectif de l'algorithme SVM est de trouver un hyperplan qui sépare les données en deux classes de manière optimale.

L'algorithme Logistic Regression est une technique d'apprentissage supervisé utilisée pour la classification. Il s'agit d'un modèle linéaire qui utilise une fonction de régression logistique pour prédire la probabilité d'appartenance à une classe particulière en fonction des valeurs d'entrée. L'objectif de l'algorithme est de trouver les coefficients optimaux de la fonction de régression logistique pour minimiser l'erreur de classification.

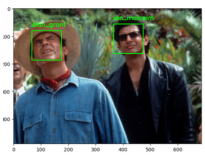
Nous avons ensuite évalué leurs performances sur l'ensemble de test, en termes de précision et de rapidité d'exécution. Pour ce faire, nous avons créé une fonction model_evaluation qui retourne la précision et le temps moyens d'un modèle sur un certain nombre de tests (ici 500).

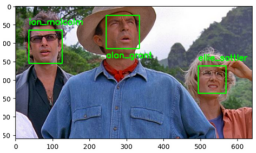
Voici les résultats :

```
Precision de Knearest: 0.99
 Temps execution moyen: 0.0121
Precision de svm: 0.97
 Temps execution moyen: 0.0046
Precision de Logistic regression: 0.97
 Temps execution moyen: 0.0158
```

Les 3 modèles atteignent une grande précision, toute supérieure ou égale à 97%.
Cependant, le modèle KNN est un peu plus performant que les autres.
Au niveau du temps d'exécution, le modèle SVM est beaucoup plus rapide.

Nous avons choisi le classificateur KNN sur les images et vidéos de test :





Le modèle reconnaît bien les visages.

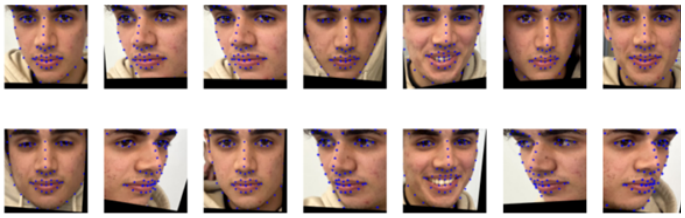
E - Personal dataset

Dans cette partie, nous avons construit un ensemble de données personnalisé, avec des images de nous et de nos amis.
En effet, depuis le debut du projet, nous utilisons un ensemble de données pré-organisé, où les images étaient déjà rassemblées et labelisées.

Notre dataset contient entre 60 à 80 photos de 8 personnes différentes prises dans divers conditions de lumière, d'angle et d'émotion.
Nous avons au total plus de 500 images.

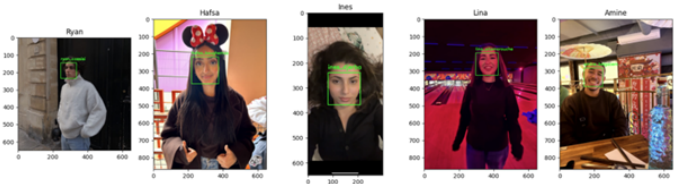
Puis nous avons appliqué le pipeline développé à chacune des étapes précédentes afin de construire notre propre système de reconnaissance faciale personnalisé.
En commençant par l'extraction des visages sur chaque image.

Nous avons ensuite aligner les visages :



Puis nous avons procédé à l'encodage de ceux-ci.

La séparation des Training et Test set est la même que précédemment (70%-30%).
On entraîne notre fully-connected network et le test sur de nouvelles images :



F - Extra : Bias analysis

Un biais dans l'apprentissage automatique se produit lorsqu'un modèle est entraîné sur des données qui ne représentent pas l'ensemble de la population cible de manière équitable, ce qui entraîne des prédictions inexactes pour certains groupes.

Dans notre contexte de la reconnaissance faciale, un biais peut se produire lorsque les données d'entraînement ne représentent pas de manière équitable la diversité ethnique, le sexe, l'âge ou d'autres caractéristiques importantes de la population.

Les biais sont un problème dans l'apprentissage automatique car ils peuvent conduire à des résultats discriminatoires et injustes.
Une situation où le biais peut être un problème est lors de l'identification des criminels. Si le modèle est entraîné sur des données majoritairement blanches, il pourrait être moins précis dans l'identification des suspects non blancs.

Afin de vérifier si notre modèle avait un biais en faveur d'un groupe spécifique, nous avons décidé de diversifier notre dataset personnel avec des ethniques de différentes personnes. On observe que notre modèle ne se trompe pas sur un groupe spécifique,et donc ne possède pas de biais. En effet, notre base se compose d'autant de femmes que d'hommes, dont 2 sur les 8 personnes ayant une peau de couleur noir.
Afin de détecter les biais potentiels, nous pouvons penser à ajouter des statistiques telles que la précision de l'identification pour les différents groupes ethniques qui peuvent être calculées.

Conclusion

Ce projet a été très enrichissant et passionnant. Nous avons pu voir une application concrète du Deep Learning et comprendre les 4 étapes clés derrière la reconnaissance faciale. Bien que les réseaux de neurones permettent aujourd'hui de réaliser un grand nombre de tâches, nous avons vu qu'un pré-traitement des données est primordial pour les mener à bien et obtenir de très bons résultats.
Il y a également une réelle réflexion à avoir lors de la création des couches du réseau.
De plus, nous avons pu appliquer les concepts vus lors des 6 TPs précédents (Les ConvNets, les Fully-connected network, la Data augmentation, les Dropout layers, la L2-norm penalization…) pour faire face à l'un des problèmes les plus récurrents : l'overtfitting.
Enfin, appliquer la reconnaissance faciale à notre propre base de données a été une manière ludique de synthétiser tous ces concepts et de réaliser le potentiel des réseaux de neurones.

References

- [1] https://direct.mit.edu/jocn/article/3/1/71/3025/Eigenfaces-for-Recognition
- [3]https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf
- [4]https://medium.com/@appstud/lab-appstud-reconnaissance-de-visages-grace-au-machine-learning-85a612b368db
- [5] https://viso.ai/deep-learning/pose-estimation-ultimate-overview/
- [6] https://medium.com/codex/face-recognition-25f7421a2268
- [7] https://www.ibm.com/fr-fr/topics/knn